Machine Learning Flow

Кстати вот он про это подробнее:
[Nuts and Bolts of Applying Deep Learning]

Ошибка на train

маленькая

большая

underfitting

- Более мощную модель
- Больше ресурсов для тренировки
- Другой подход

Ошибка на val

маленькая

большая

overfitting

- Больше данных
- Больше регуляризации
- Другой подход

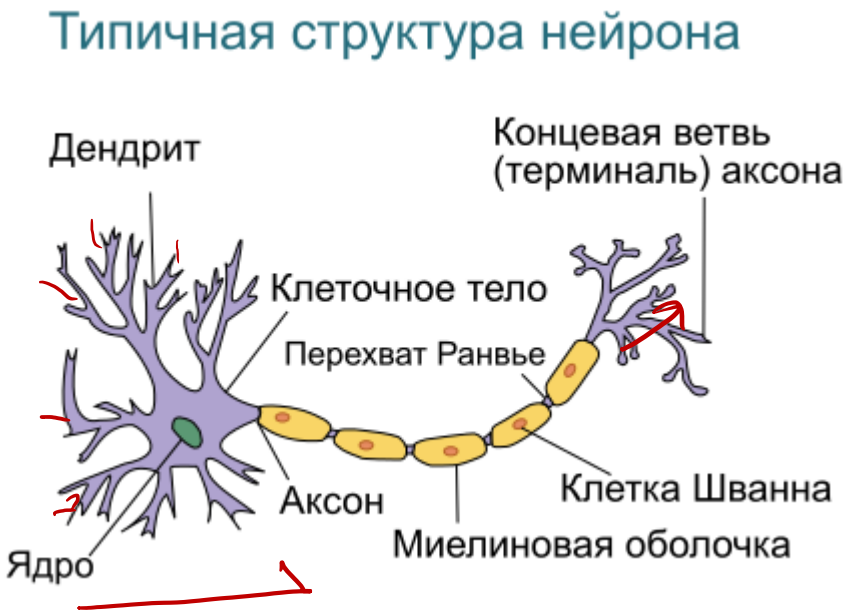
Ошибка на test

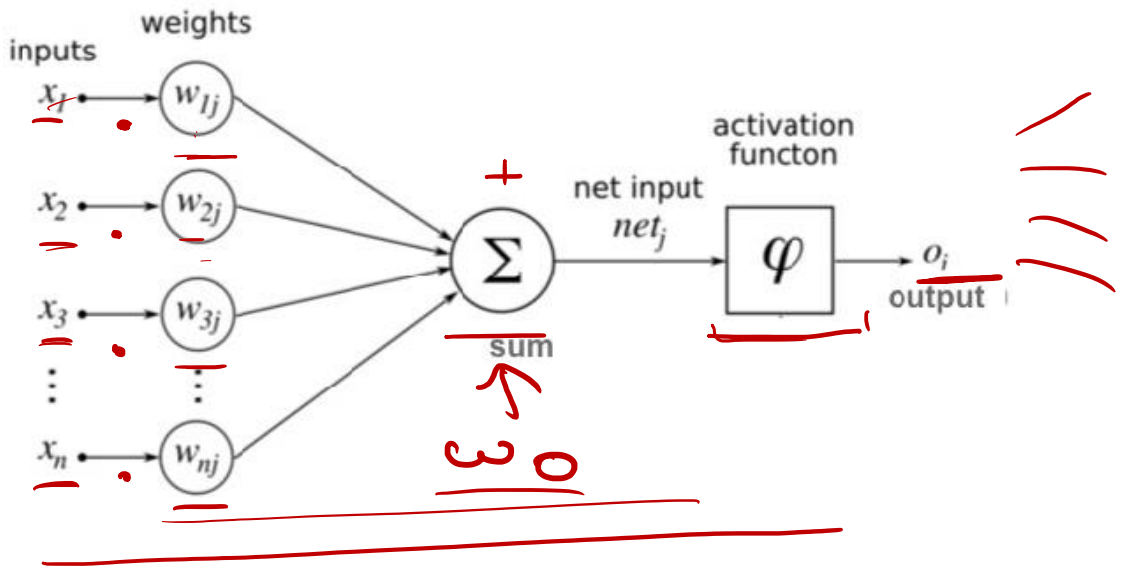
маленькая

большая

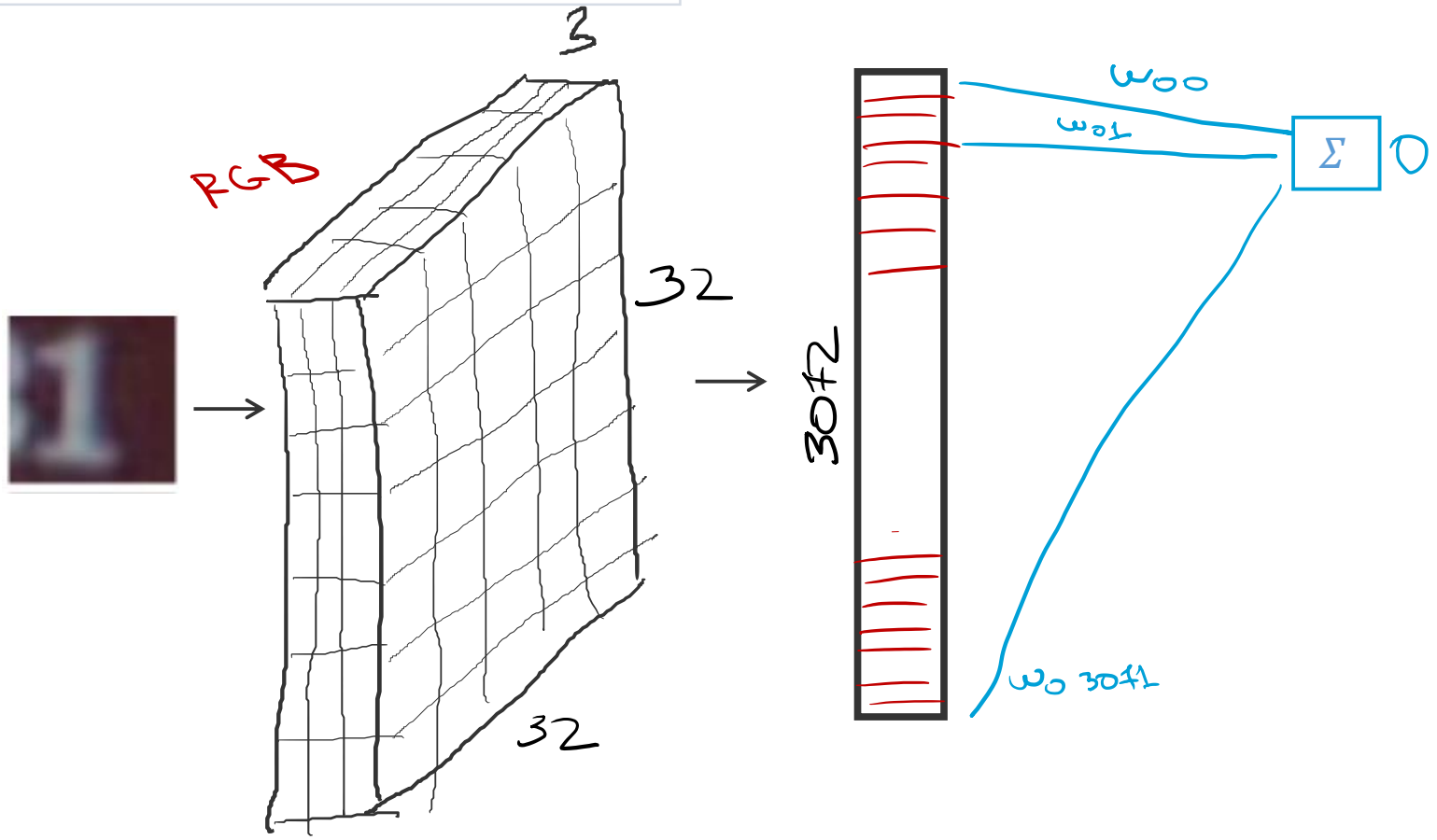
- Отличаются train и test
- Больше данных, таких как test

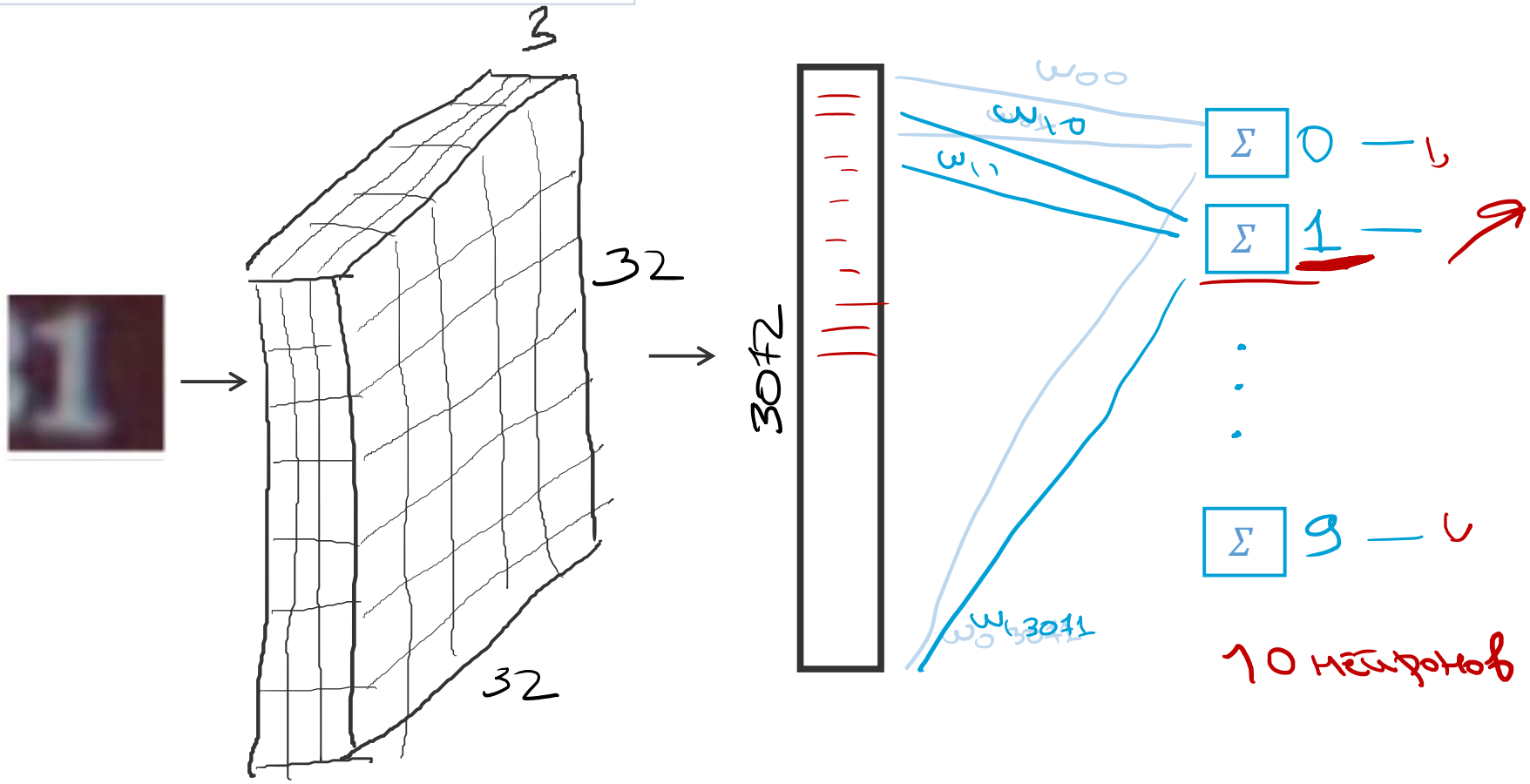
Нейронная сеть
Neural network

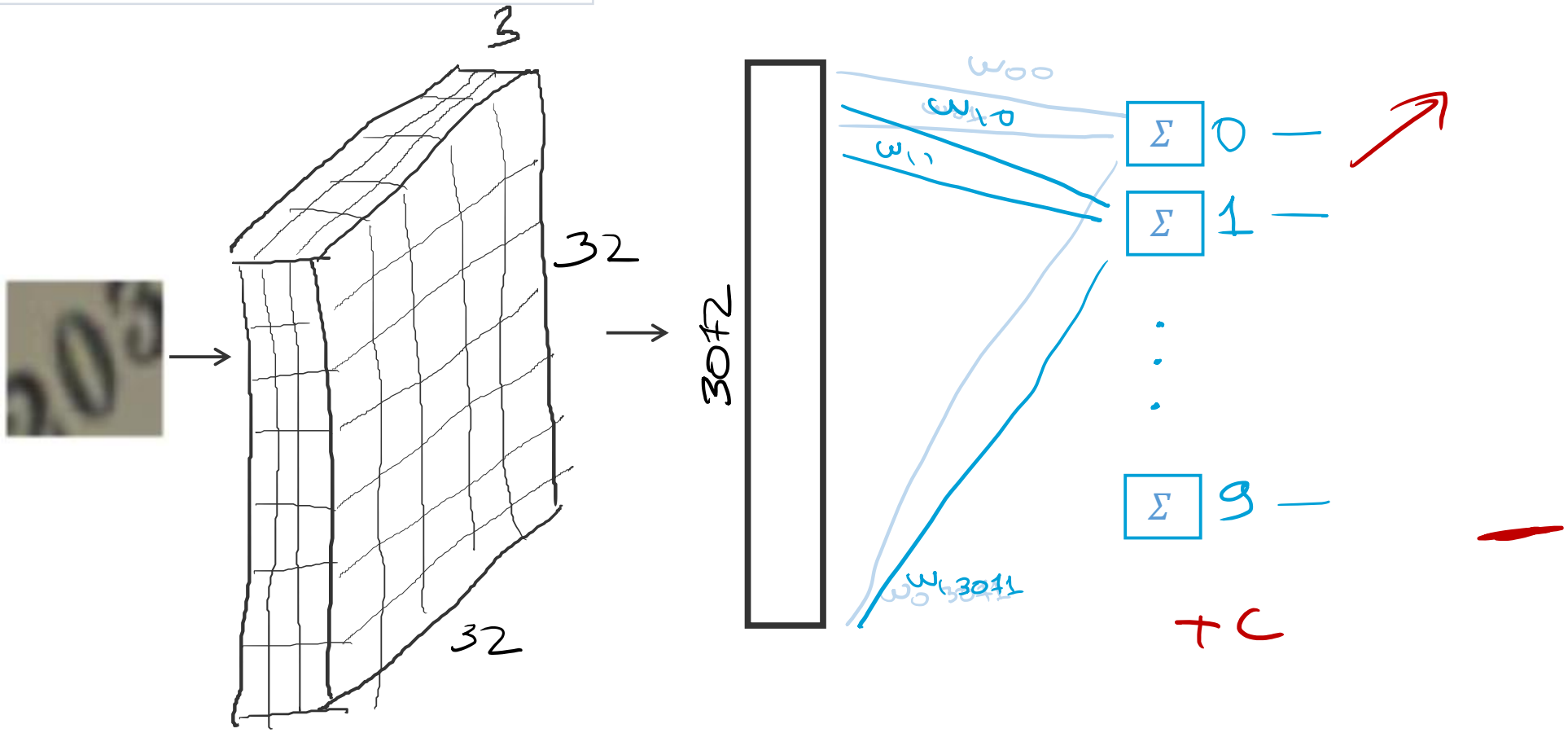




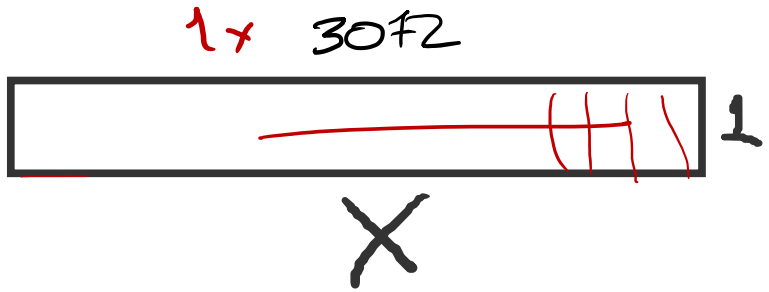
[Wikipedia]

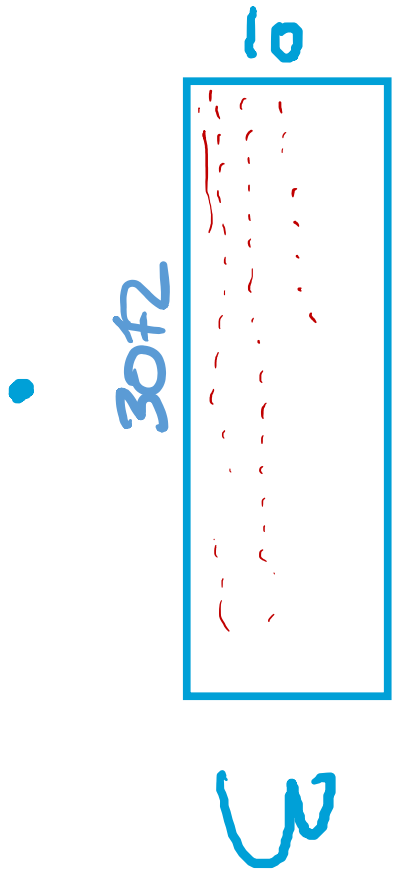

3072
3
RGB
32
32
3072
w_{00}
w_{01}
Σ
0
$w_{0\ 3071}$

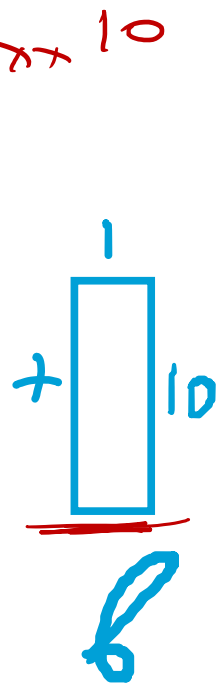

3
32
32
3072
w_{00}
w_{10}
w_{11}
w_{13071}
Σ
0
1
9
10 нейронов

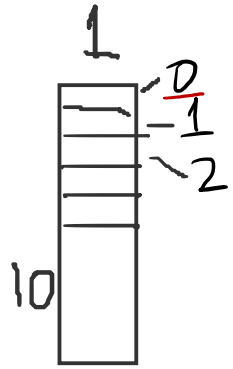

3
32
32
3072
w_{00}
w_{10}
w_{11}
w_{13071}
Σ 0
Σ 1
Σ 9
+c

Линейный классификатор
Linear classifier



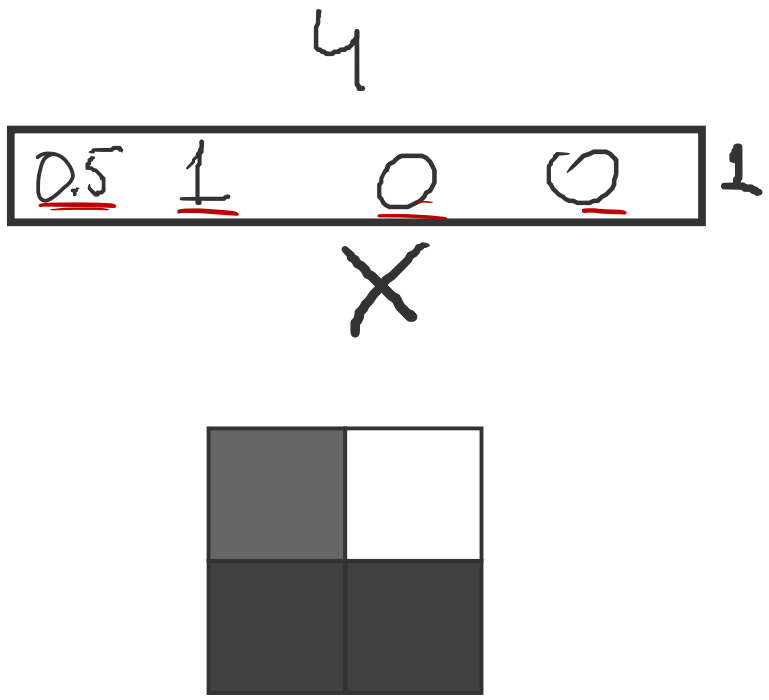


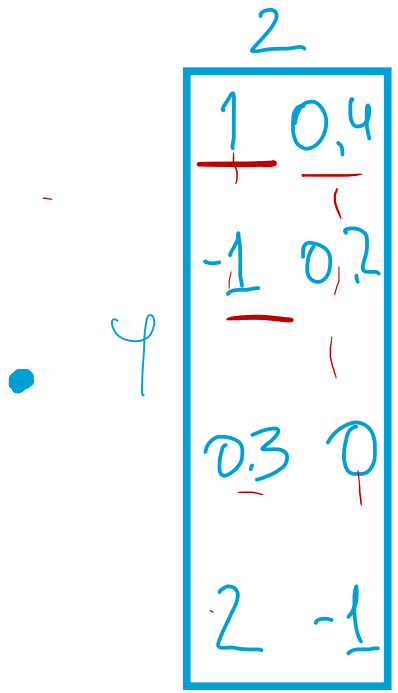


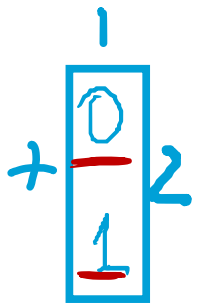


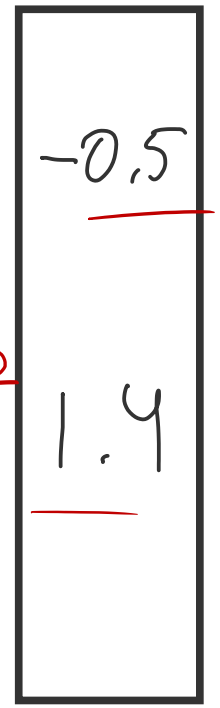
$$y = x \cdot w + b$$

Линейный классификатор
Linear classifier



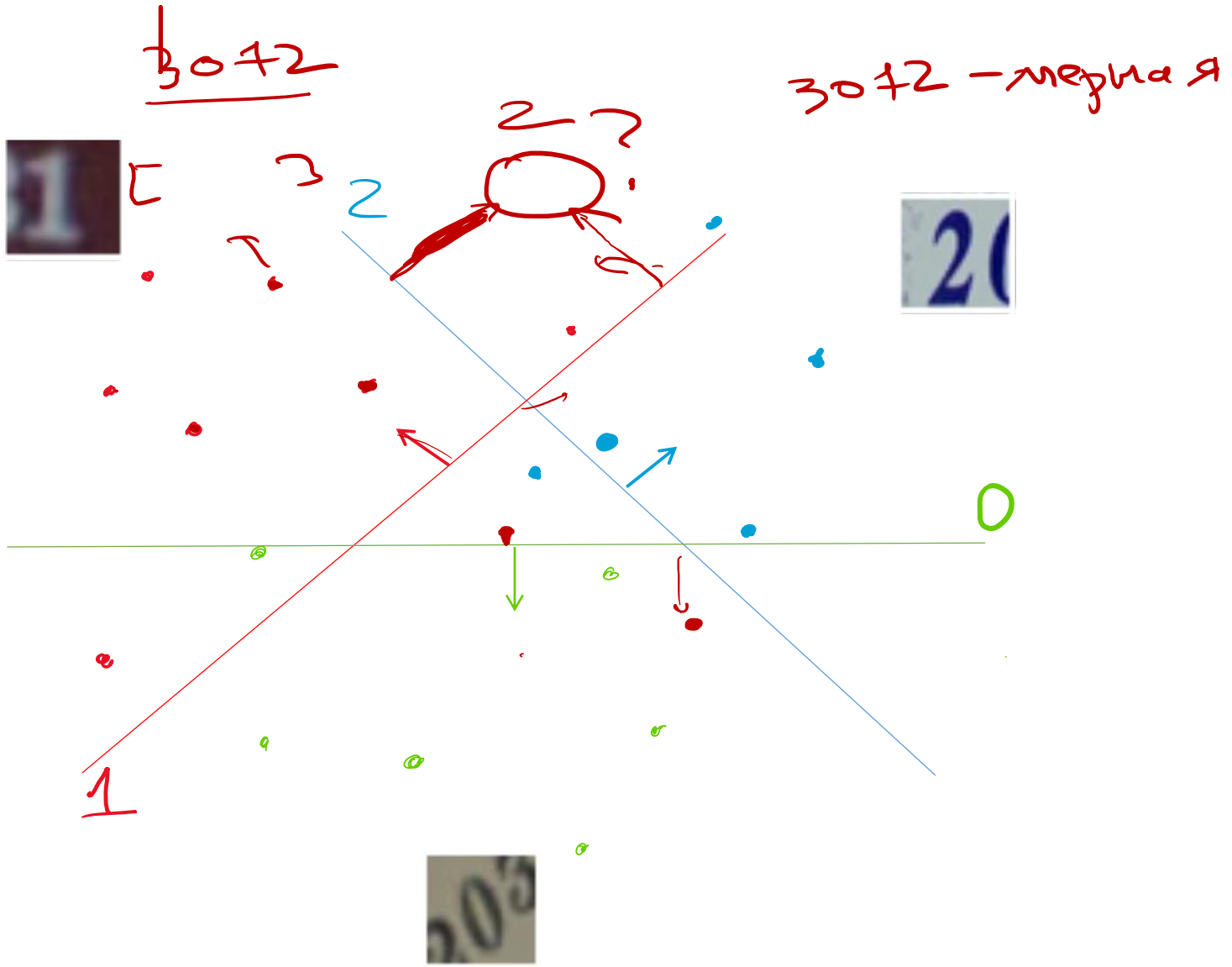




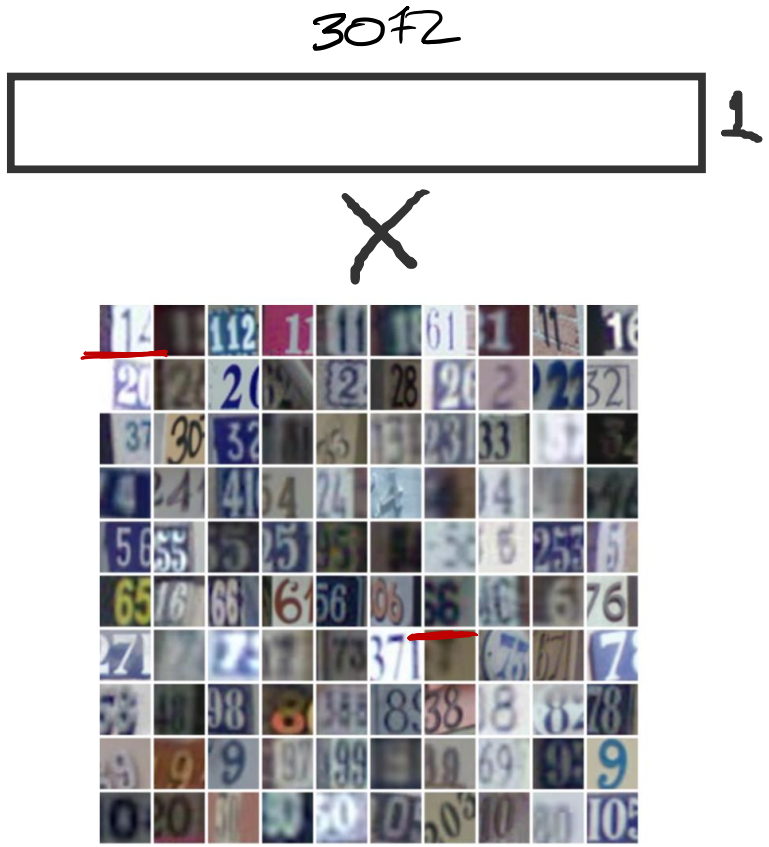


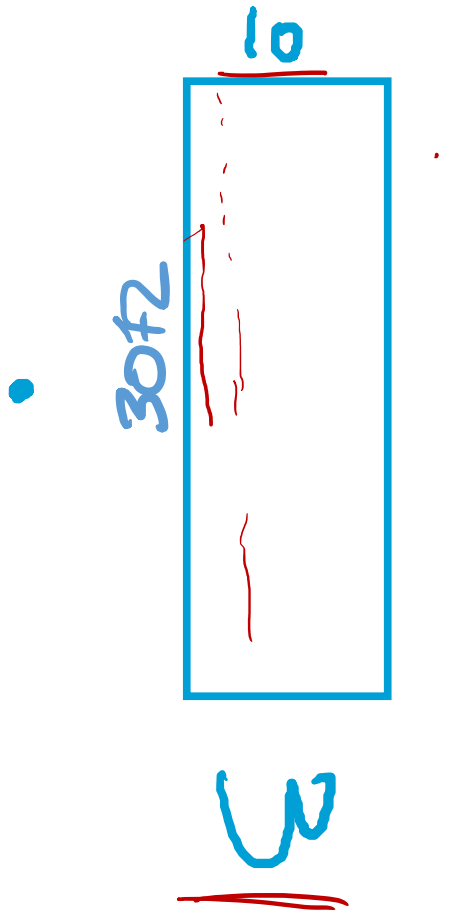
multi-class

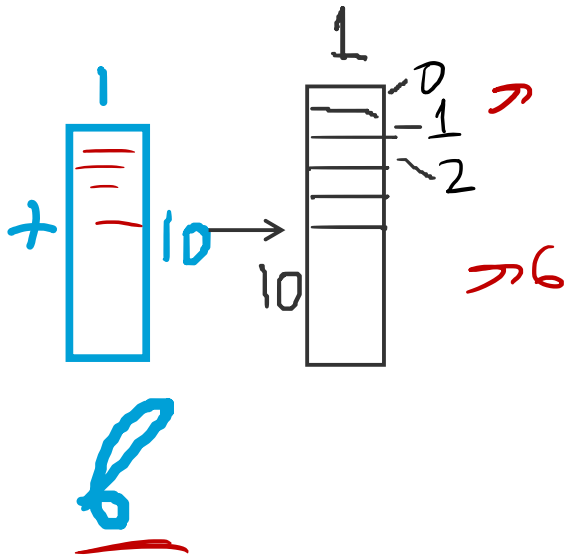
Разделяющие плоскости



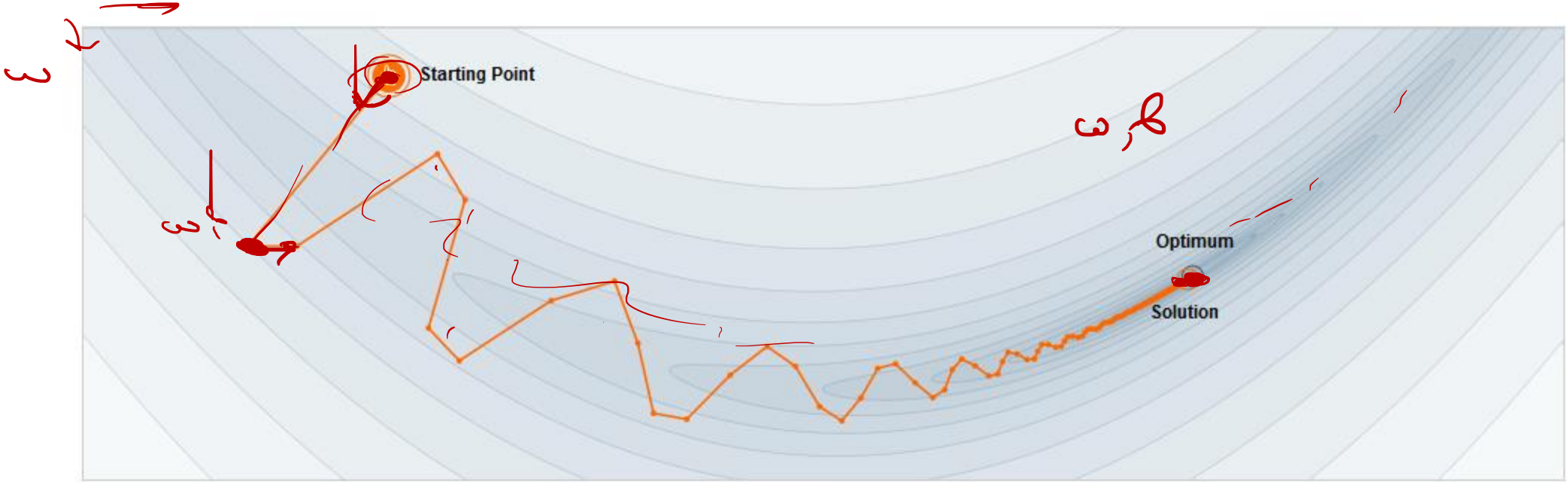
Найти лучшие w и b







Градиентный спуск
Gradient descent

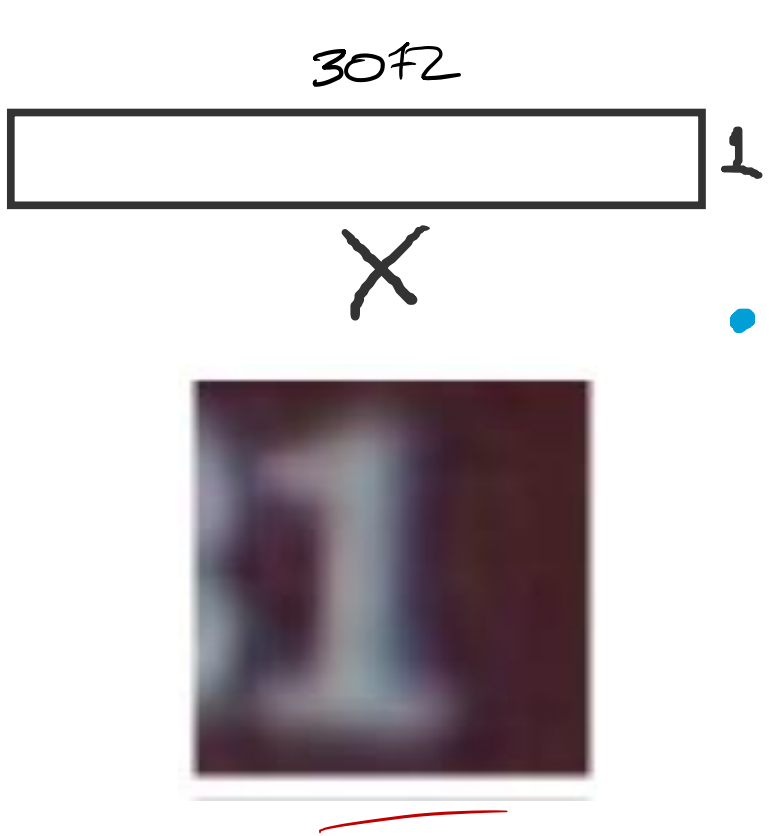


distll.pub

Softmax

$$p(C = 0|x) = \frac{e^{y_0}}{e^{y_0} + e^{y_1} + \cdots + e^{y_n}} = \frac{e^{y_0}}{\sum_i e^{y_i}}$$

$$p(C = 1|x) = \frac{e^{y_1}}{e^{y_0} + e^{y_1} + \cdots + e^{y_n}} = \frac{e^{y_1}}{\sum_i e^{y_i}}$$

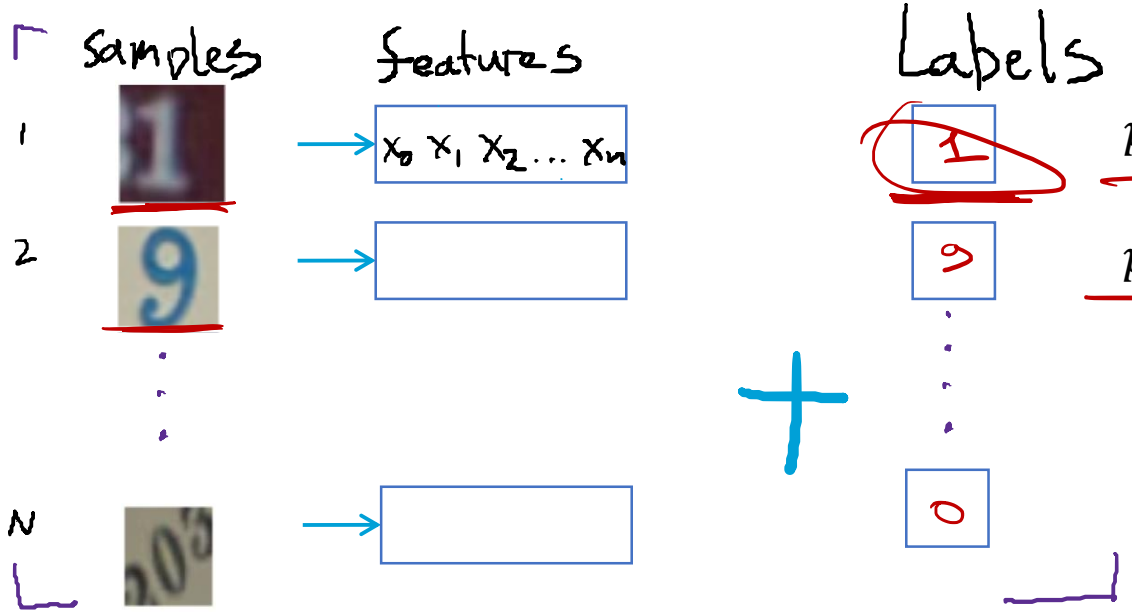


Принцип максимального правдоподобия
Maximum likelihood

ground truth

training data

w, b



$$p(data) = \prod_s p(c = gt_s|x_s)$$

w, b

Negative Log-likelihood:

$$-\ln p(data) = -\sum_s \ln p(c = gt_s|x_s)$$

w, b

Cross-Entropy loss

w.r.t. w, b

$$= -\sum_s \ln \frac{e^{(wx_s+b)_{gt_s}}}{\sum_i e^{(wx_s+b)_i}} = L$$

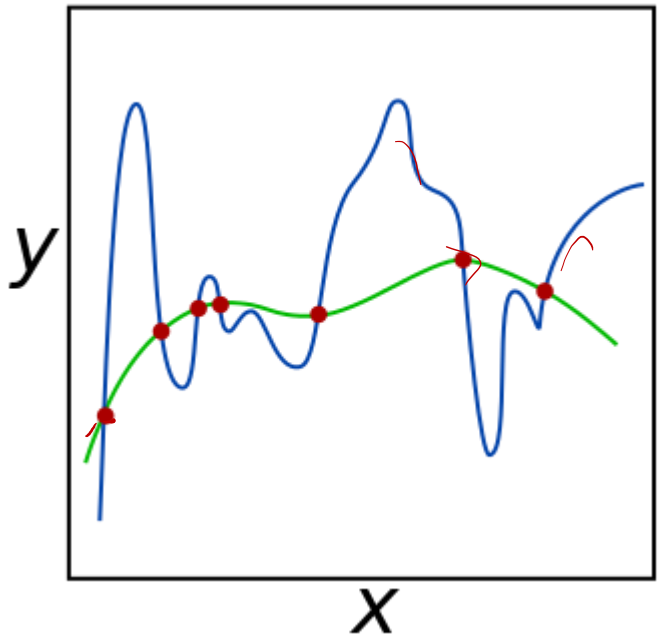
loss

s = sample

Регуляризация
Regularization

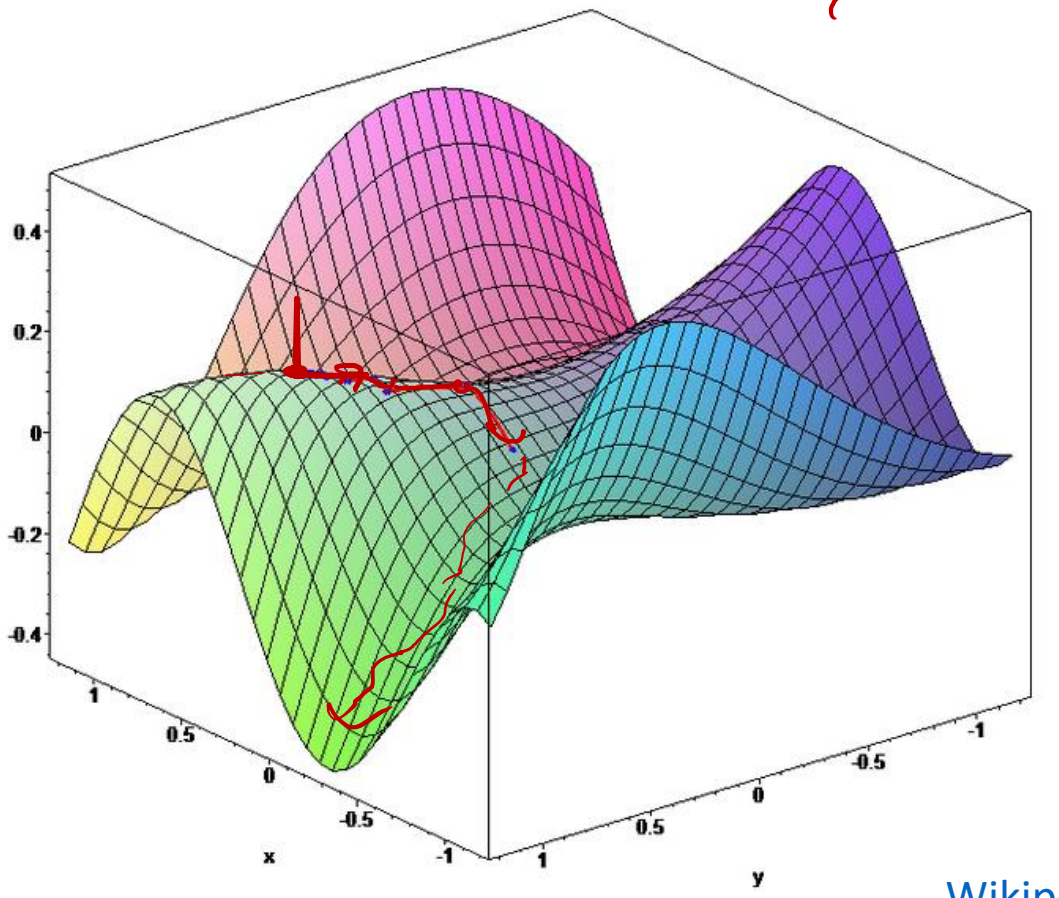
$$L = -\sum_s \ln \frac{e^{(wx_s+b)gt_s}}{\sum_i e^{(wx_s+b)_i}} + \lambda R(w,b)$$

$$R(w,b) = \|w\|_2^2 + \|b\|_2^2$$



[Wikipedia]

Gradient descent
Градиентный спуск



Wikipedia

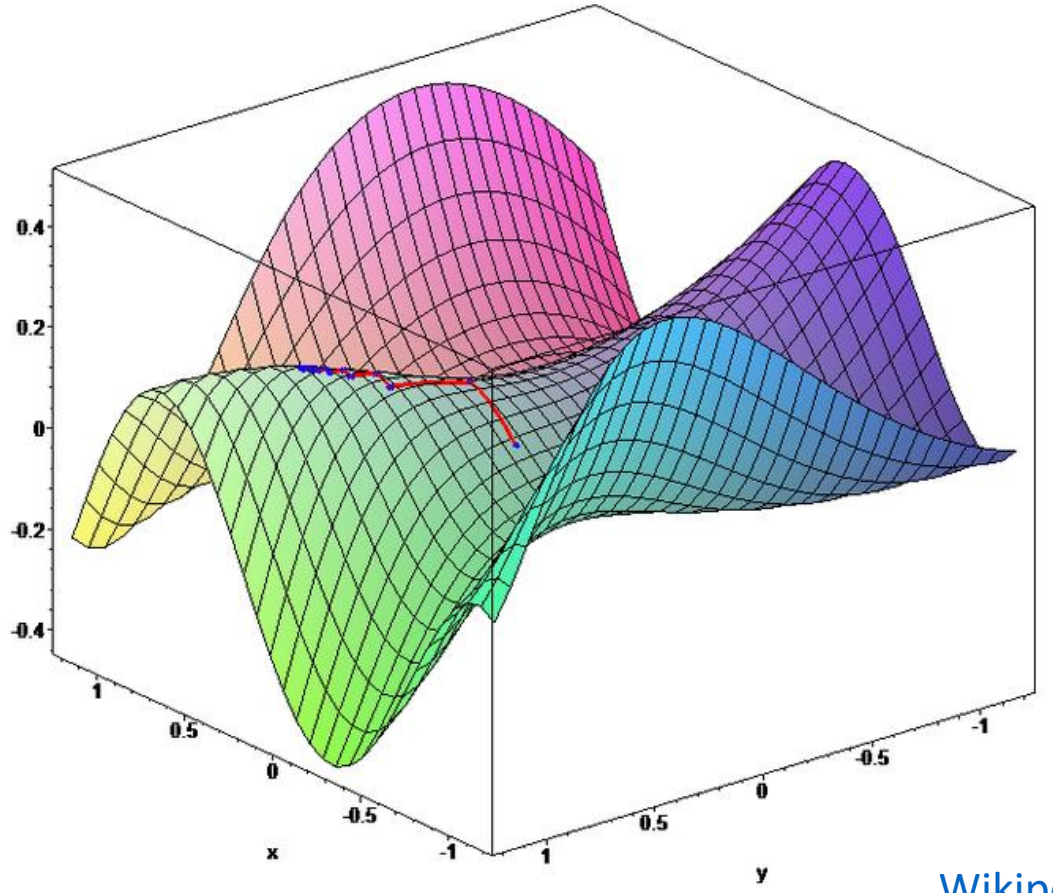
$$\vec{w} = \vec{w} - \eta \vec{\nabla}_w L$$

$$\vec{b} = \vec{b} - \eta \vec{\nabla}_b L$$

finite differences

$$L'(x) \approx \frac{L(x+\varepsilon) - L(x-\varepsilon)}{2\varepsilon}$$

Gradient descent
Градиентный спуск



$$\vec{w} = \vec{w} - \eta \overrightarrow{\nabla_w} L$$

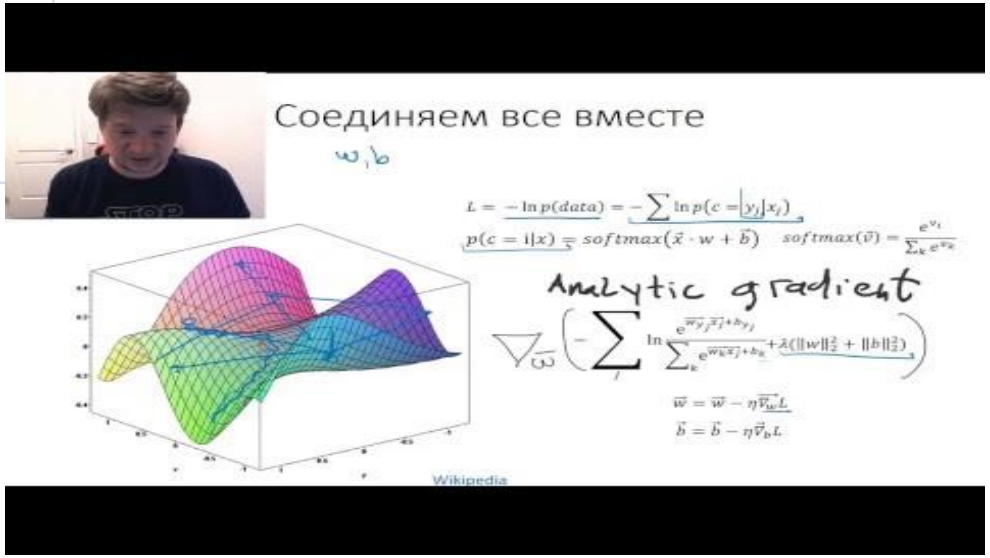
$$\vec{b} = \vec{b} - \eta \vec{\nabla}_b L$$

Analytic gradient

$$\frac{\partial}{\partial b_k}\left[-\sum_s \ln \frac{e^{(wx_s+b)_{gt_s}}}{\sum_i e^{(wx_s+b)_i}} + \lambda(\|w\|_2^2 + \|b\|_2^2) \right] =$$

Wikipedia

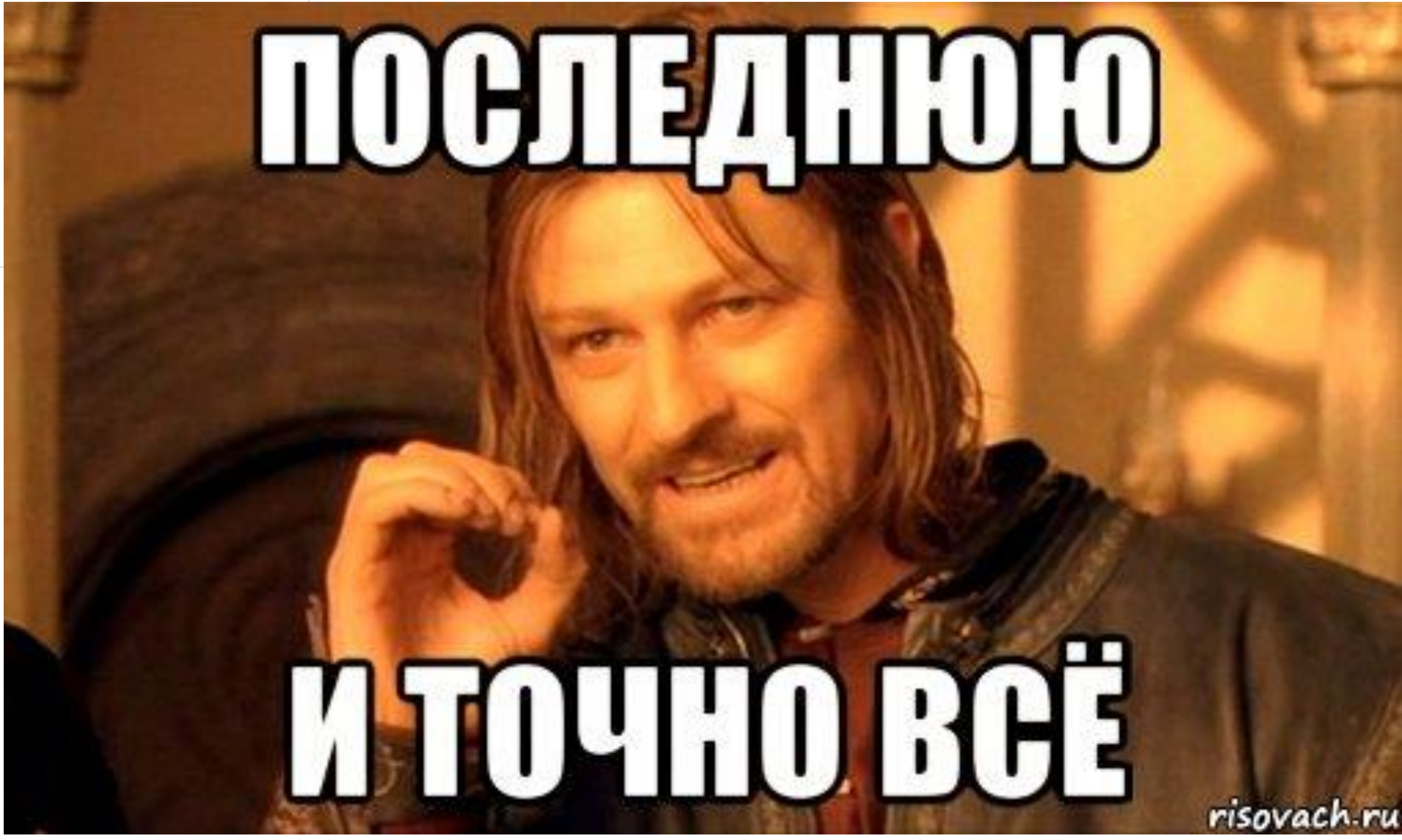
Помощь в выводе



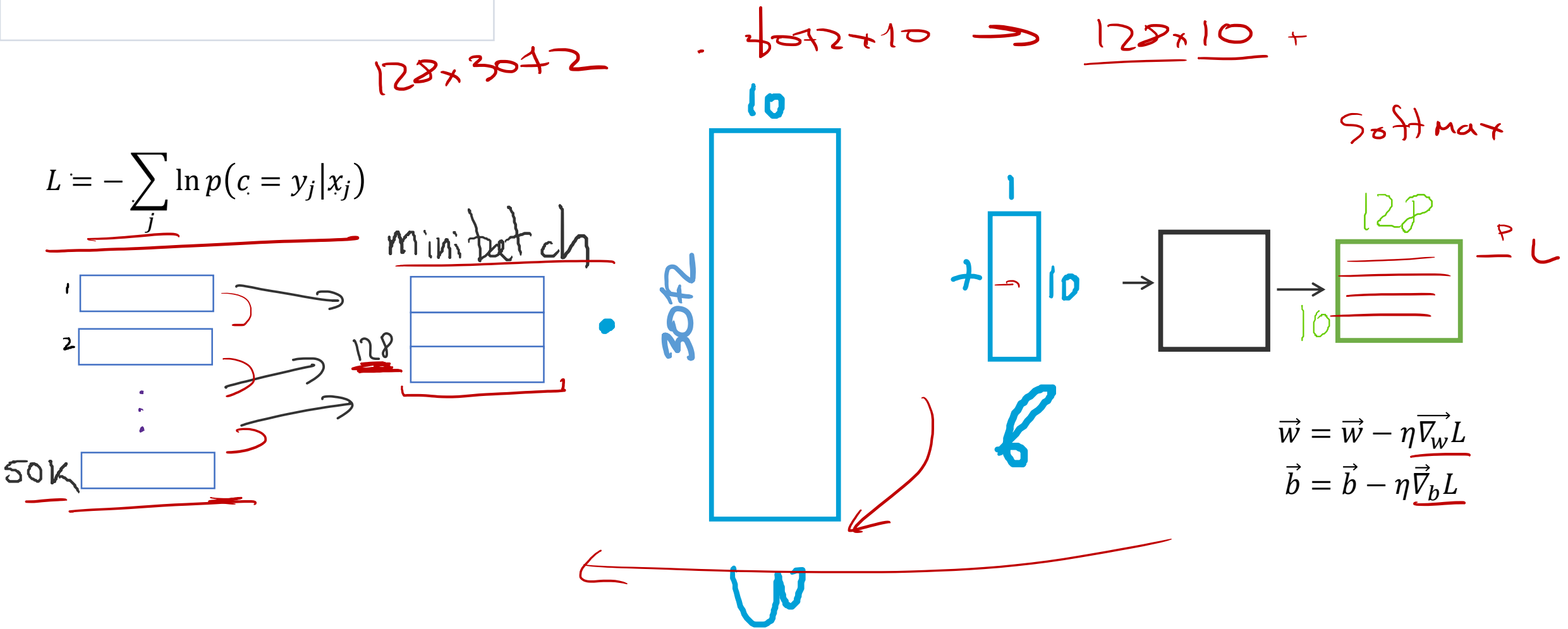
Link

http://cs231n.github.io/linear-classify/
http://cs231n.github.io/optimization-1/

методичка

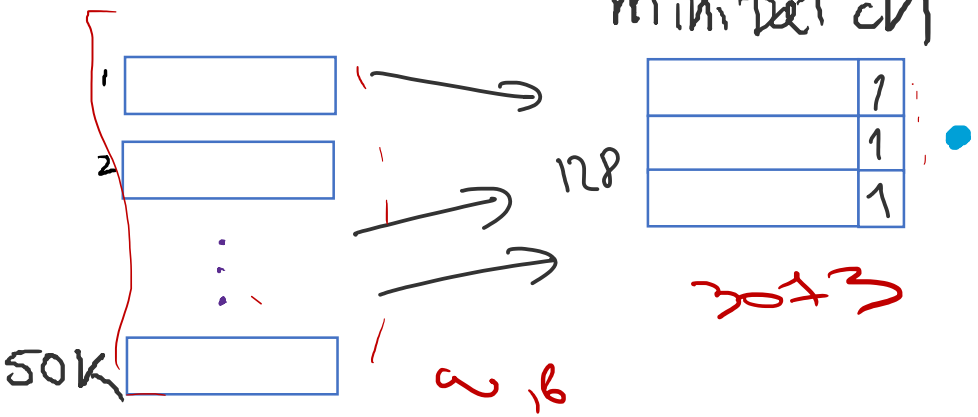

ПОСЛЕДНЮЮ
И ТОЧНО ВСЁ
risovach.ru

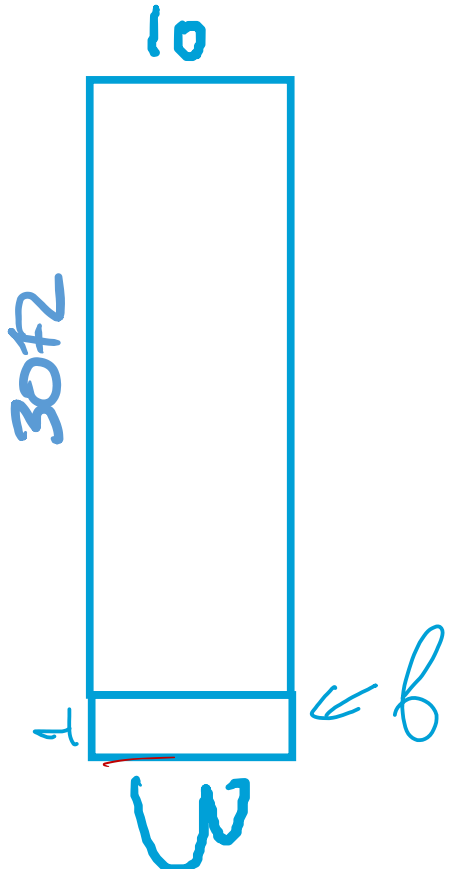
Стохастический градиентный спуск
Stochastic Gradient Descent (SGD)

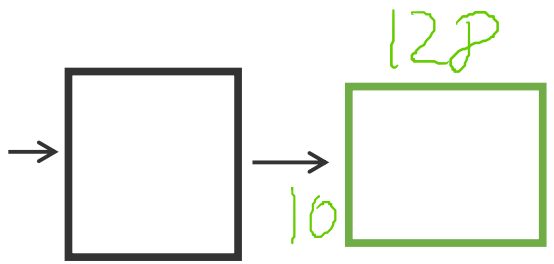


Стохастический градиентный спуск
Stochastic Gradient Descent (SGD)

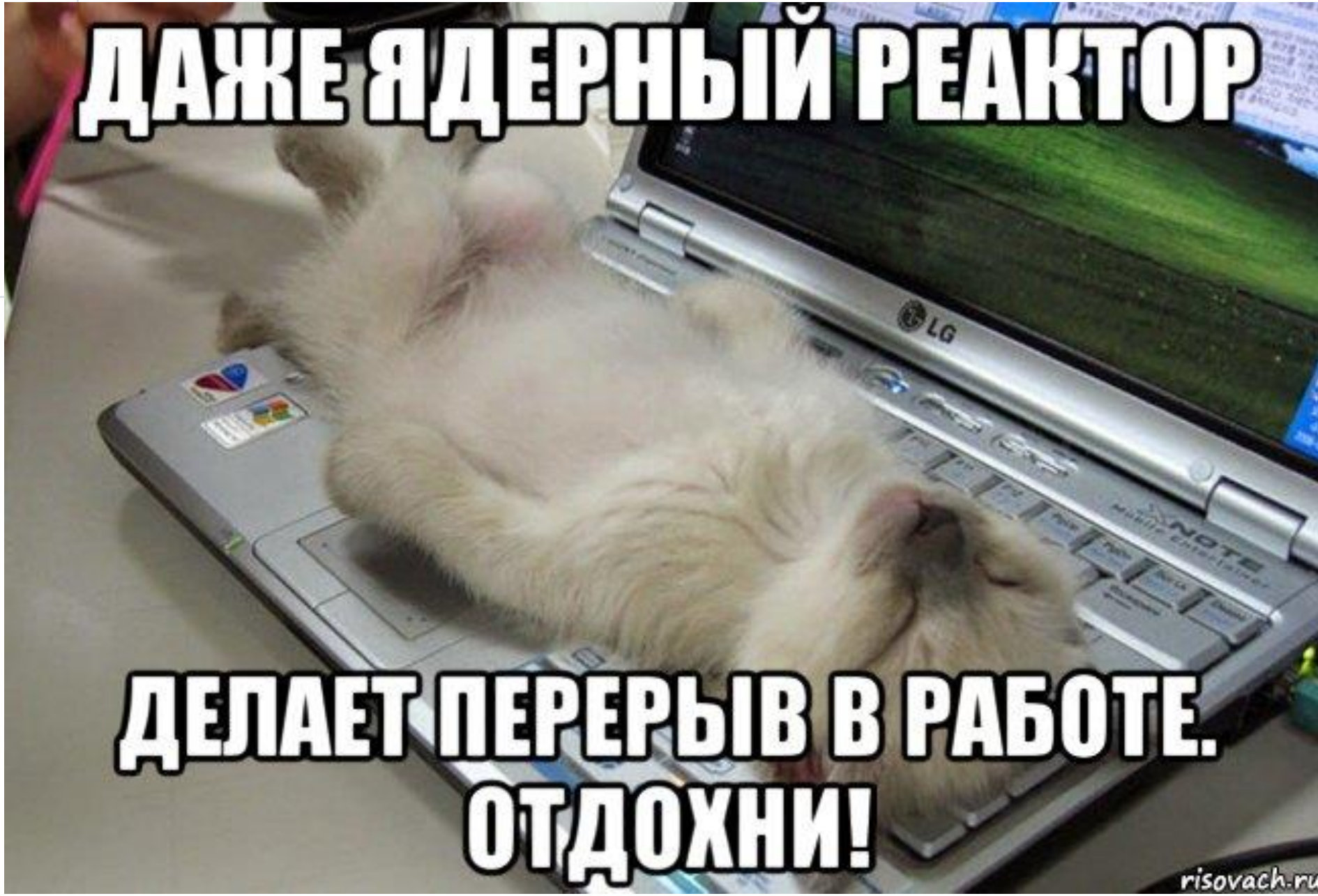
$$L = -\sum_j \ln p(c = y_j | x_j)$$



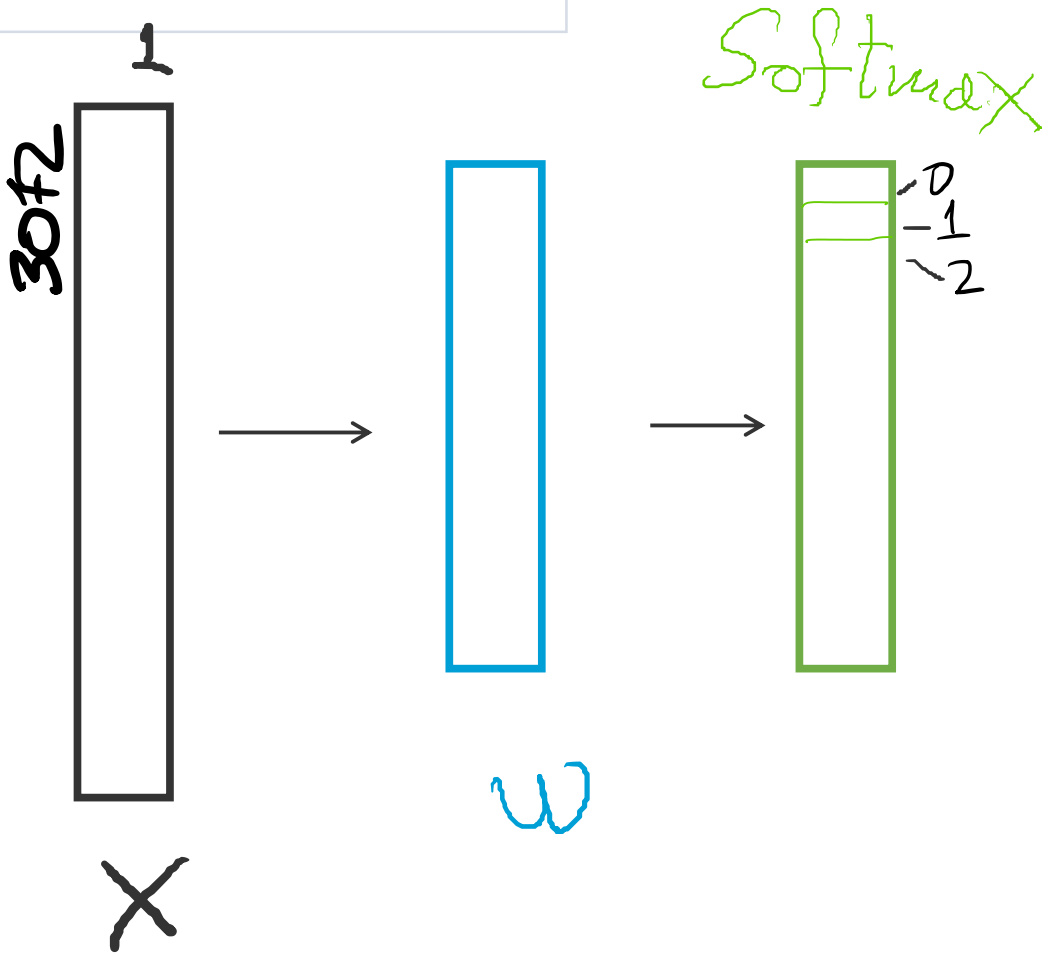


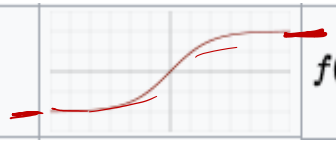
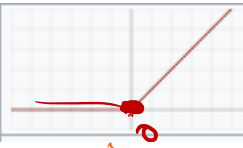


$$\vec{w} = \vec{w} - \eta \vec{\nabla}_w L$$


ДАЖЕ ЯДЕРНЫЙ РЕАКТОР
ДЕЛАЕТ ПЕРЕРЫВ В РАБОТЕ.
ОТДОХНИ!
LG
NOTE
risovach.ru

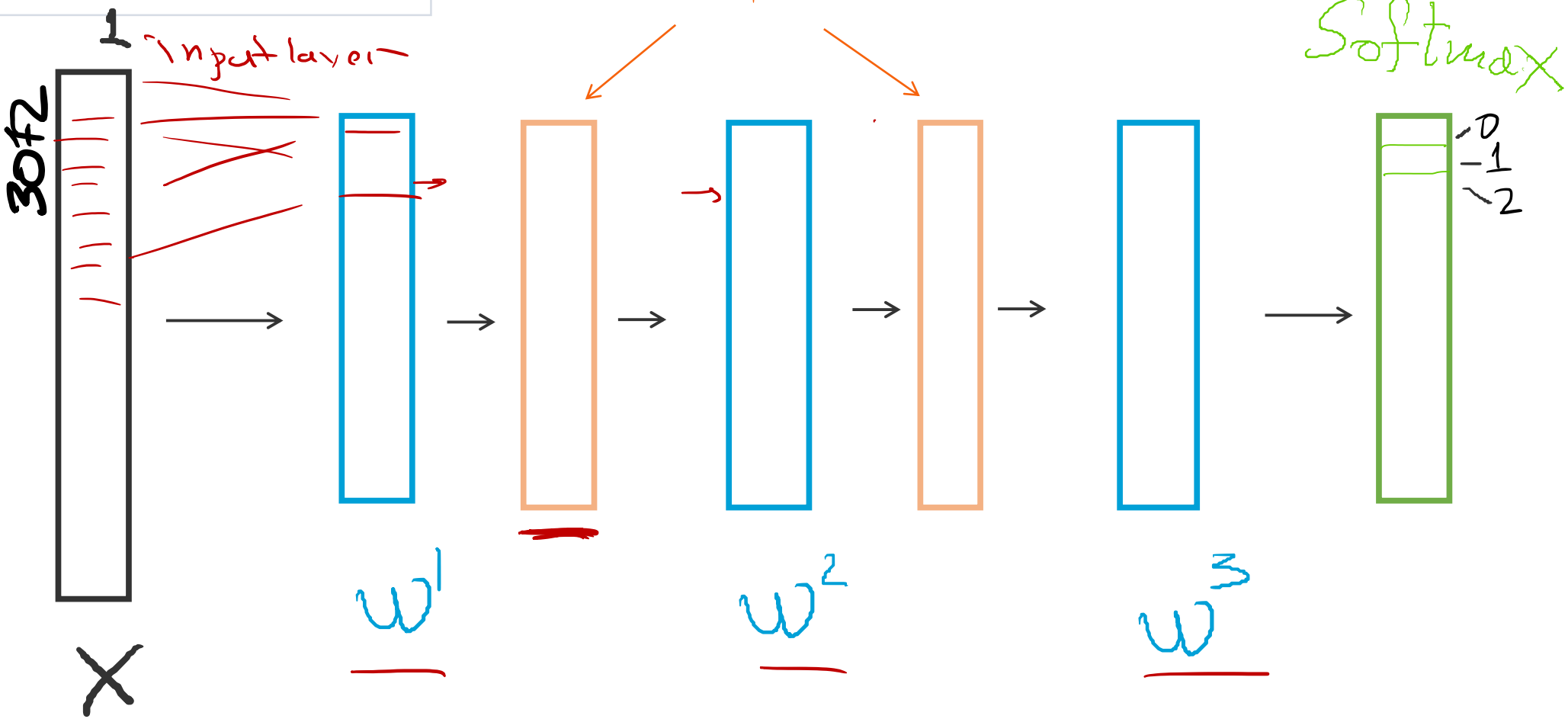
Так вот, нейронные сети



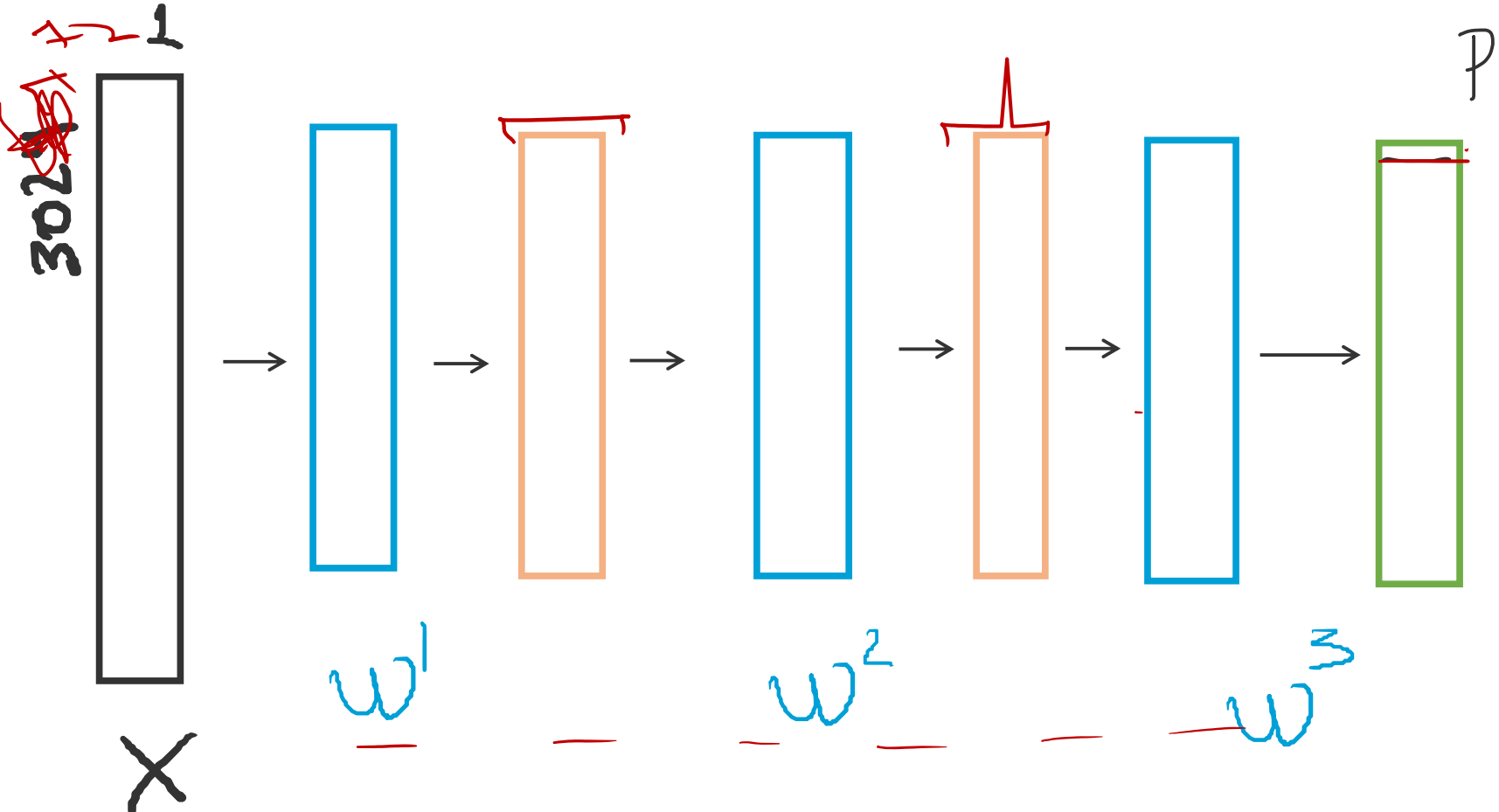
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
Rectified linear unit (ReLU)[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

non-linear function

Wikipedia



Тренировка



$$L = -\sum_j \ln p\left(c = y_j \middle| x_j\right) + \lambda R(\omega)$$

$$\overrightarrow{w^1} = \overrightarrow{w^1} - \eta \overrightarrow{\nabla_{w^1} L}$$

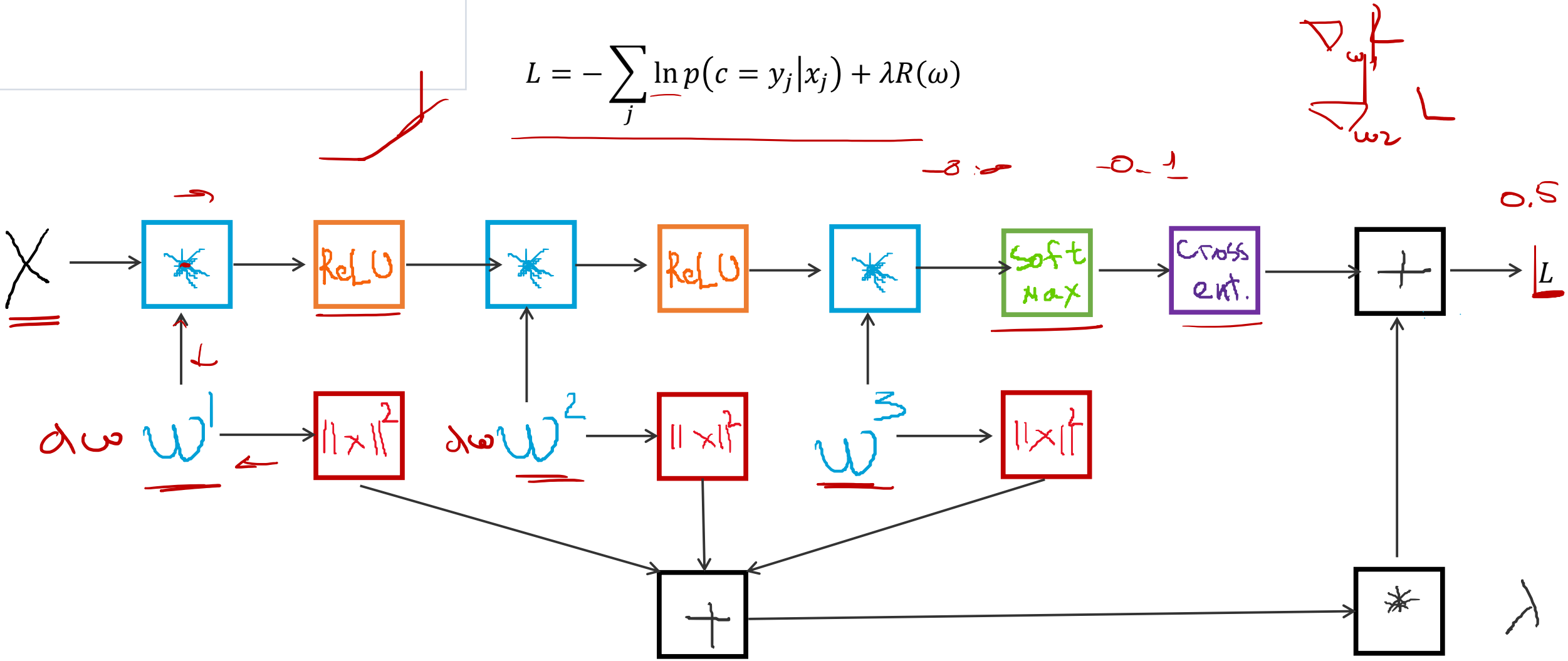
$$\overrightarrow{w^2} = \overrightarrow{w^2} - \eta \overrightarrow{\nabla_{w^2} L}$$

$$\overrightarrow{w^3} = \overrightarrow{w^3} - \eta \overrightarrow{\nabla_{w^3} L}$$

Как посчитать $\overrightarrow{\nabla_w} L$?

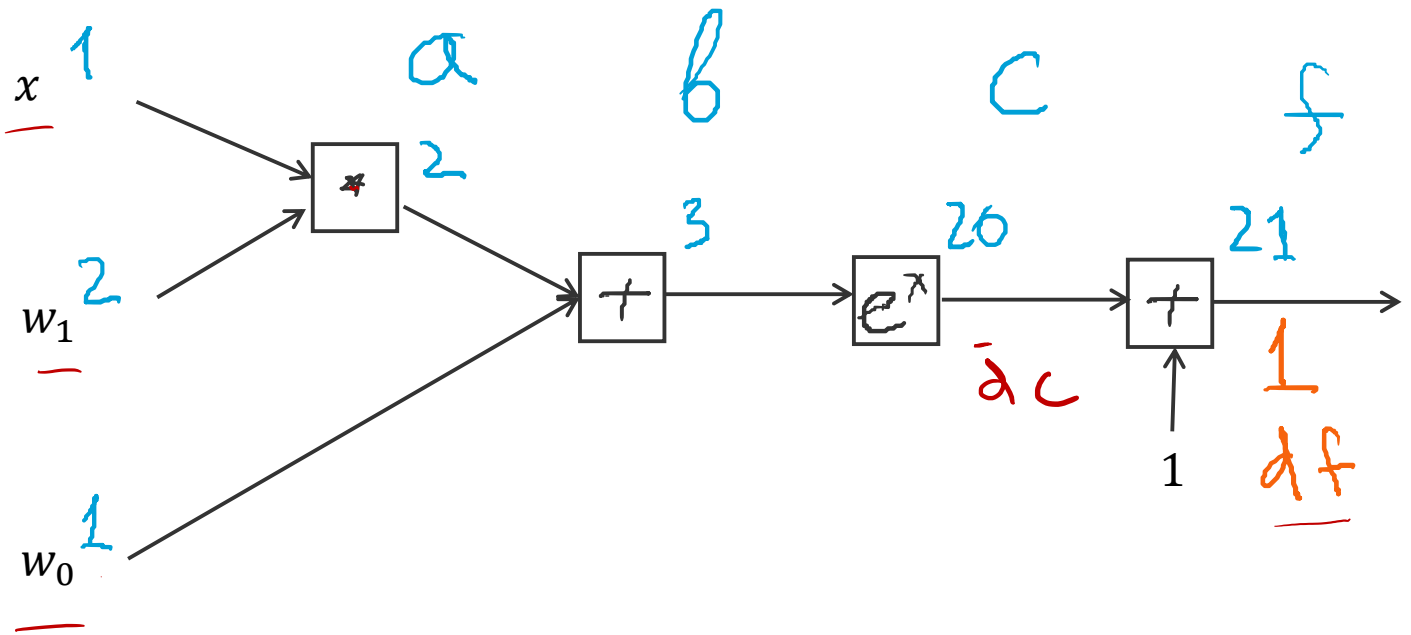
Граф вычислений
Computational graph

$$L = -\sum_j \ln p(c = y_j | x_j) + \lambda R(\omega)$$



Тренируемся на кошках

$f(g(x))$ $\quad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\quad f(x, w) = 1 + e^{w_1 x + w_0}$

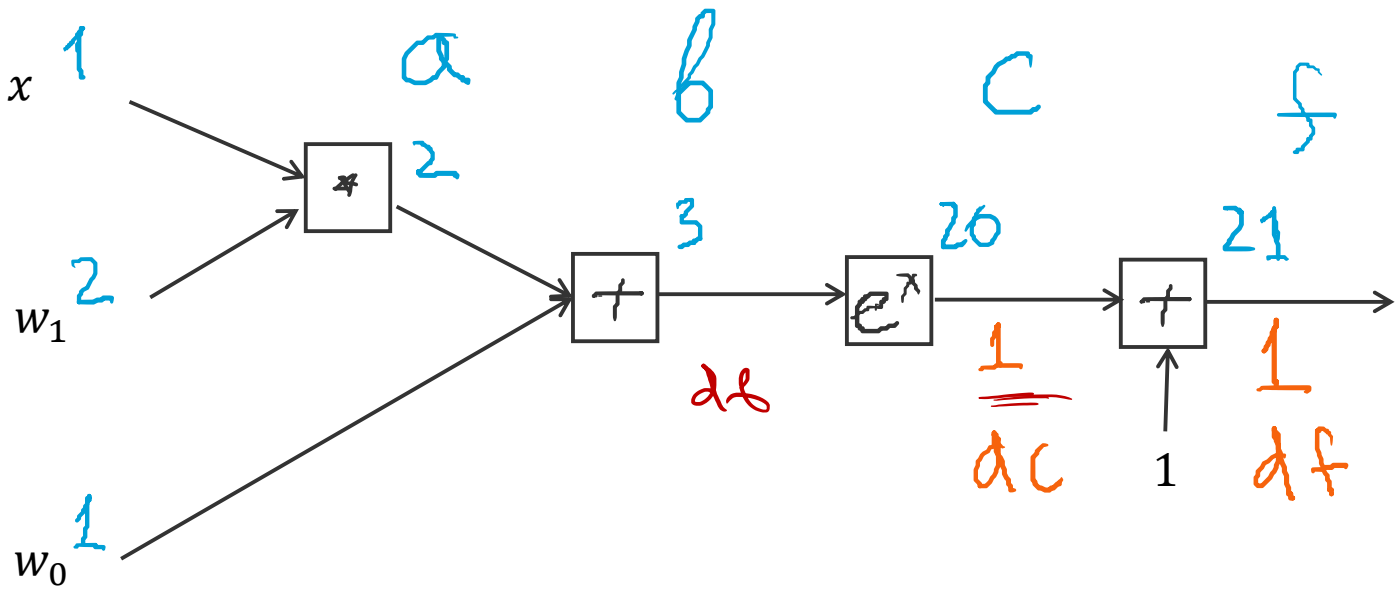


Тренируемся на кошках

$f(g(x))$ $\quad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\quad f(x, w) = 1 + e^{w_1 x + w_0}$

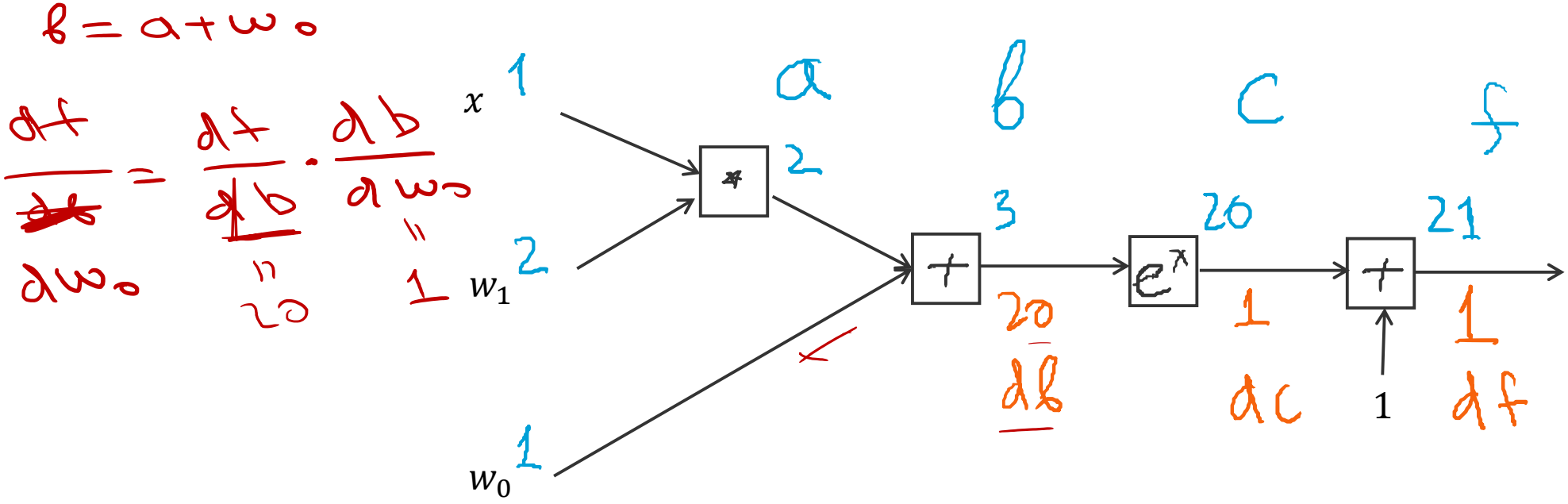
$\frac{df}{db} = \frac{df}{dc} \cdot \frac{dc}{db}$ $\quad \frac{df}{dc} = 1$ $\quad \frac{dc}{db} = e^b$

$c = e^b$



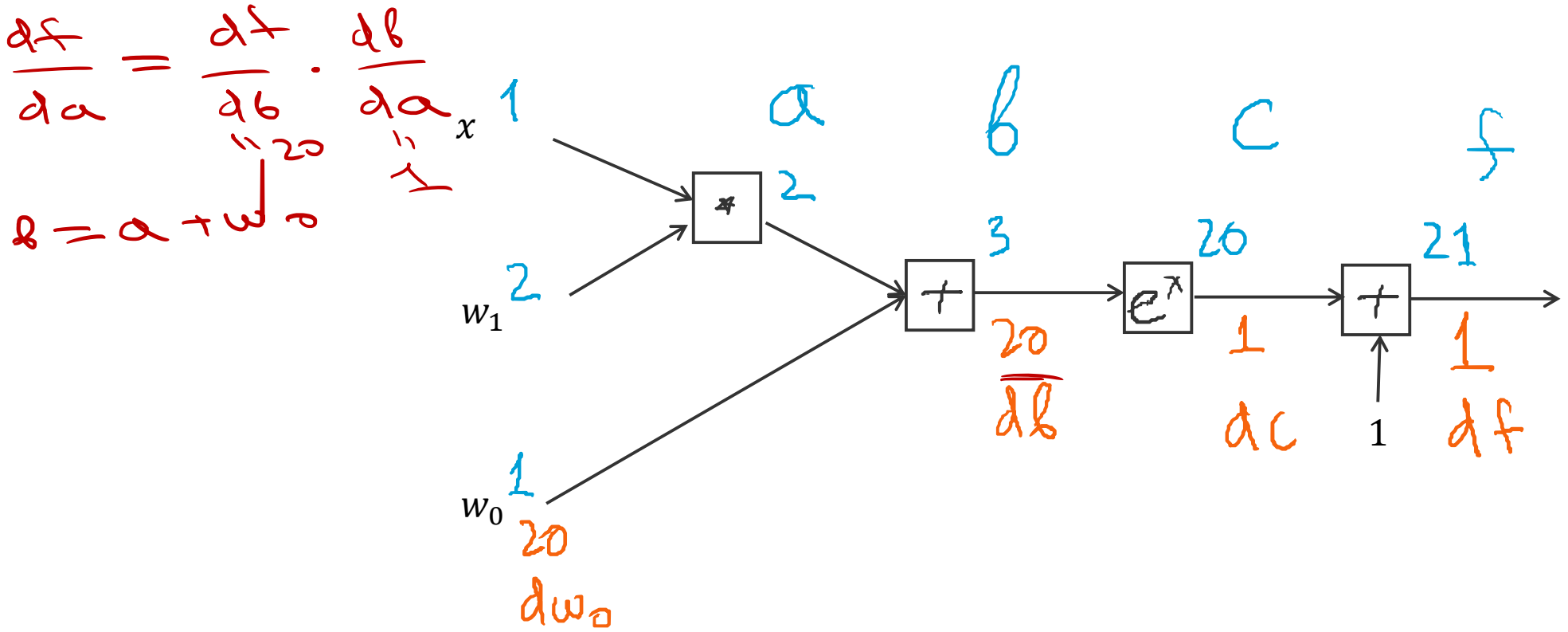
Тренируемся на кошках

$f(g(x))$ $\qquad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\qquad f(x, w) = 1 + e^{w_1 x + w_0}$



Тренируемся на кошках

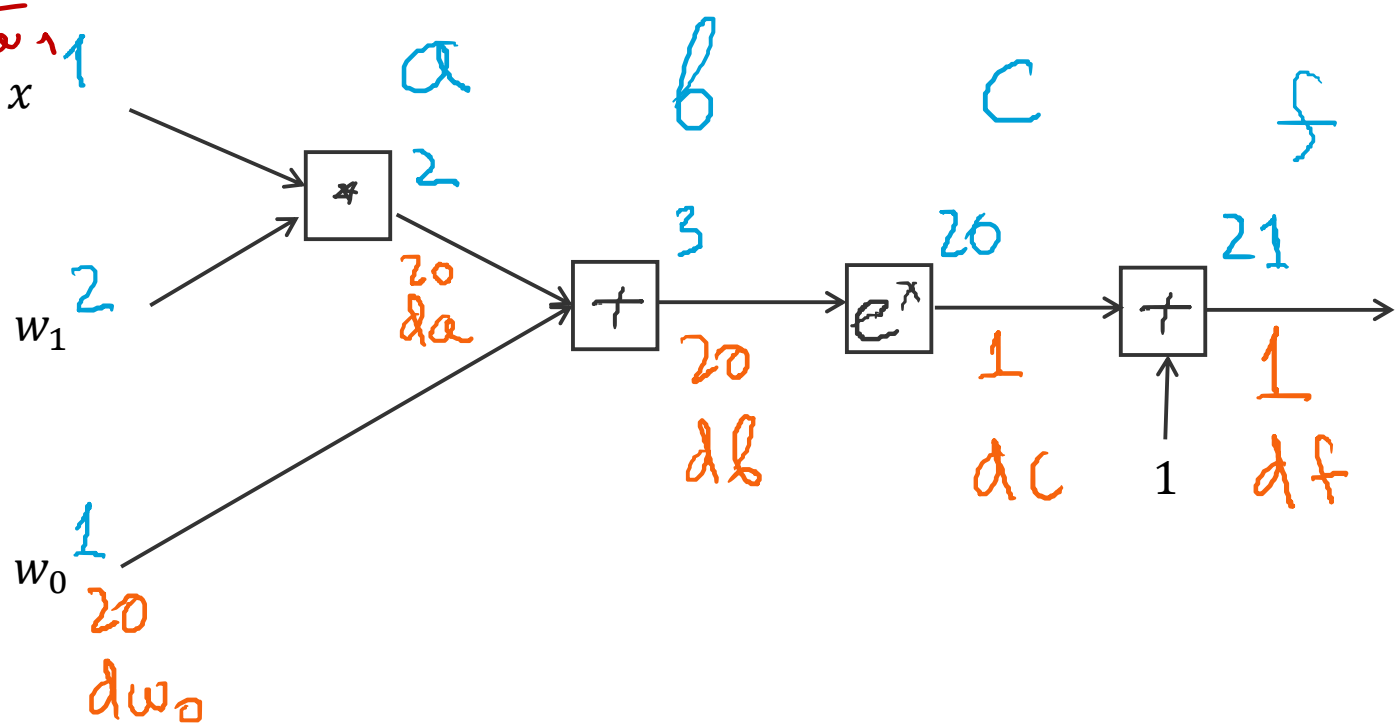
$f(g(x))$ $\quad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\quad f(x, w) = 1 + e^{w_1 x + w_0}$



Тренируемся на кошках

$f(g(x))$ $\quad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\quad f(x, w) = 1 + e^{w_1 x + w_0}$

$\frac{df}{dw_1} = \frac{df}{da} \cdot \frac{da}{dw_1}$ $\quad a = w_1 \cdot x$

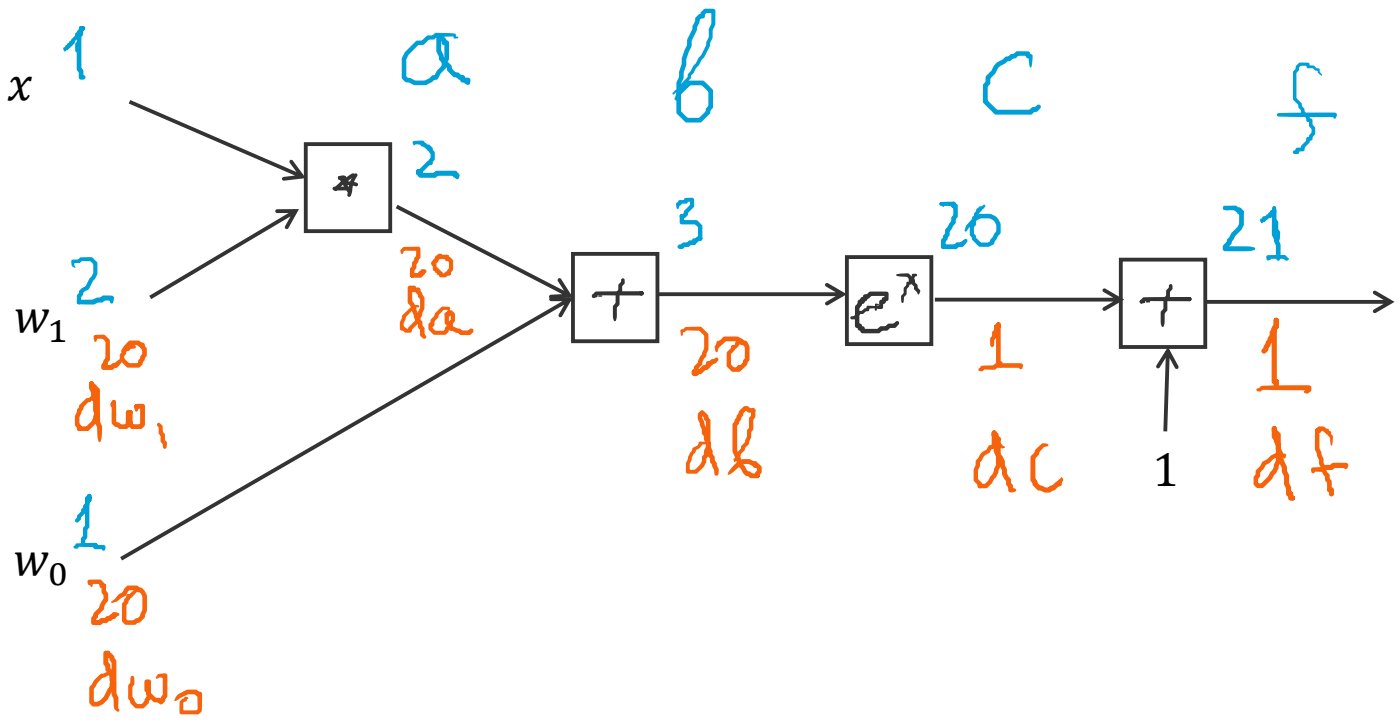


Тренируемся на кошках

$f(g(x))$ $\qquad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\qquad f(x, w) = 1 + e^{w_1 x + w_0}$

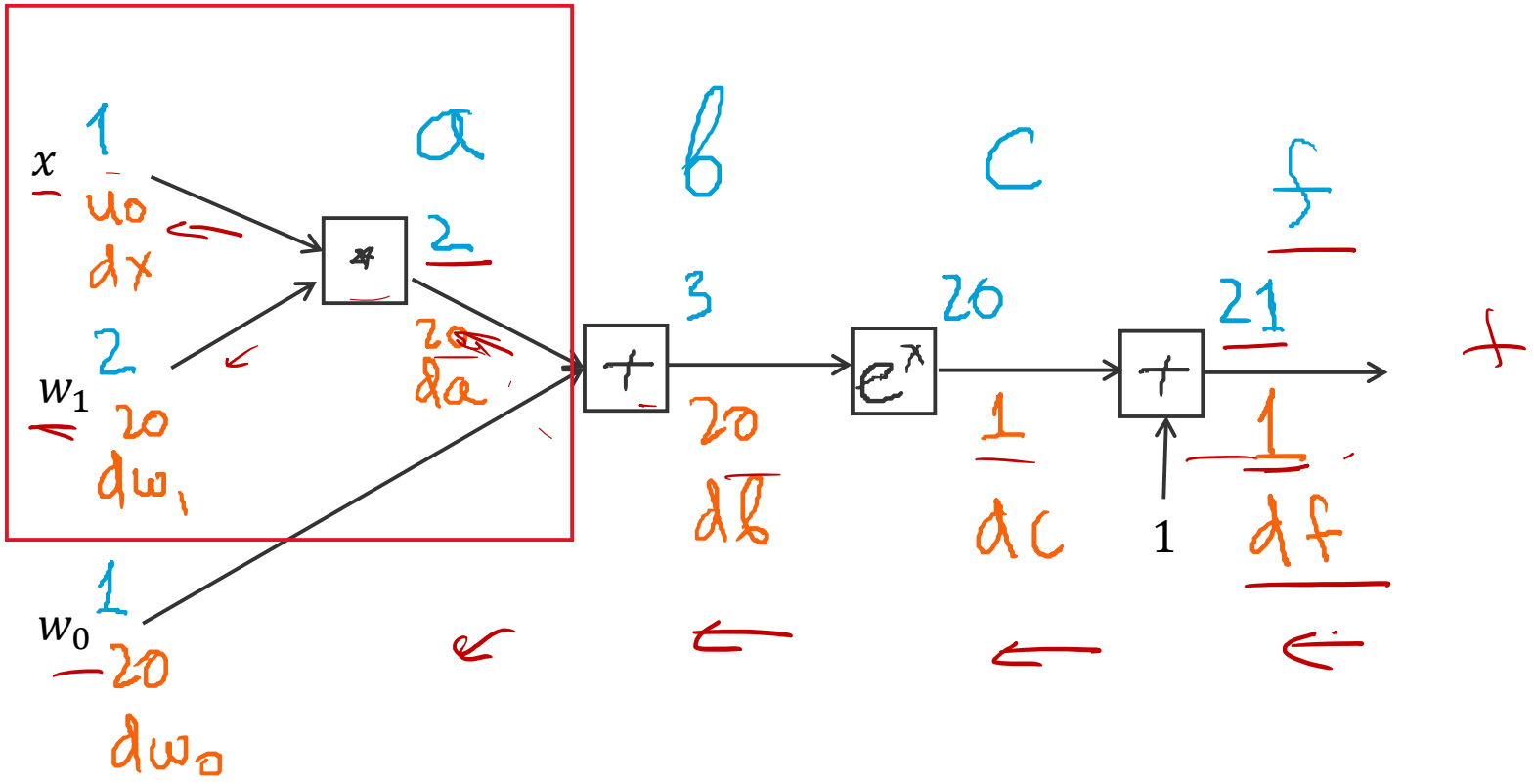
$\frac{df}{dx} = \frac{df}{da} \cdot \frac{da}{dx}$ $\quad \frac{df}{da} = 20$ $\quad \frac{da}{dx} = w_1$

$a = x \cdot w_1$

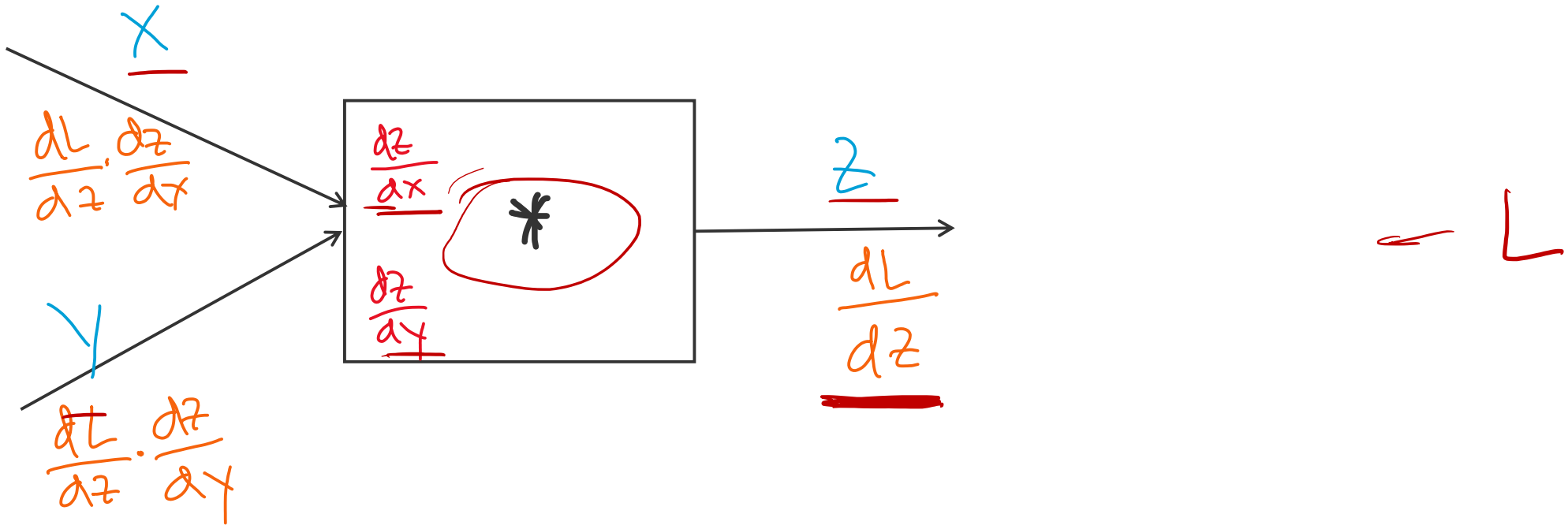


Обратное распространение ошибки
Backpropagation

$f(g(x))$ $\quad \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$ $\quad f(x, w) = 1 + e^{w_1 x + w_0}$



Общая схема вычисления градиента



В коде

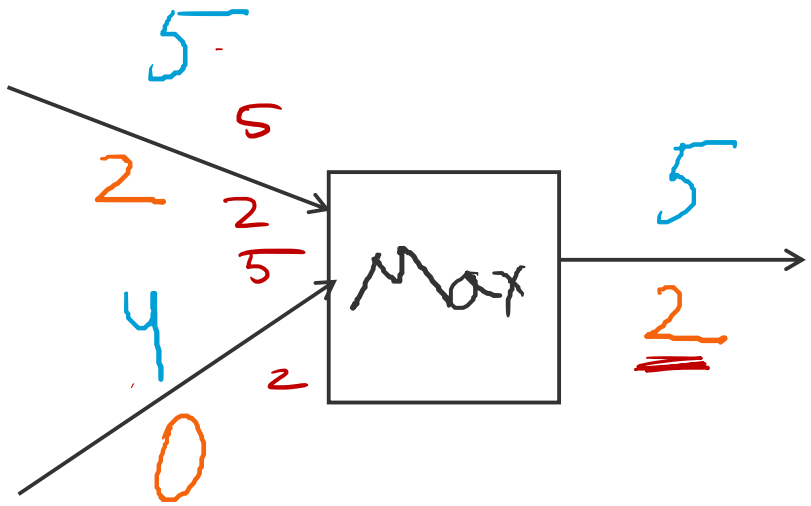
forward

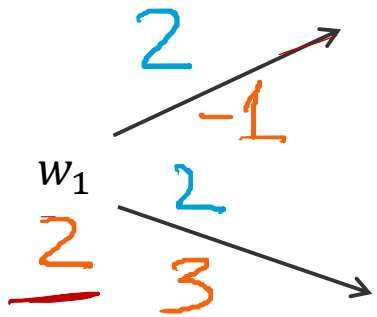
```
x = 1
w1 = 2
w0 = 1
a = x*w1
b = a + w0
c = np.exp(b)
f = 1 + c
```

backward

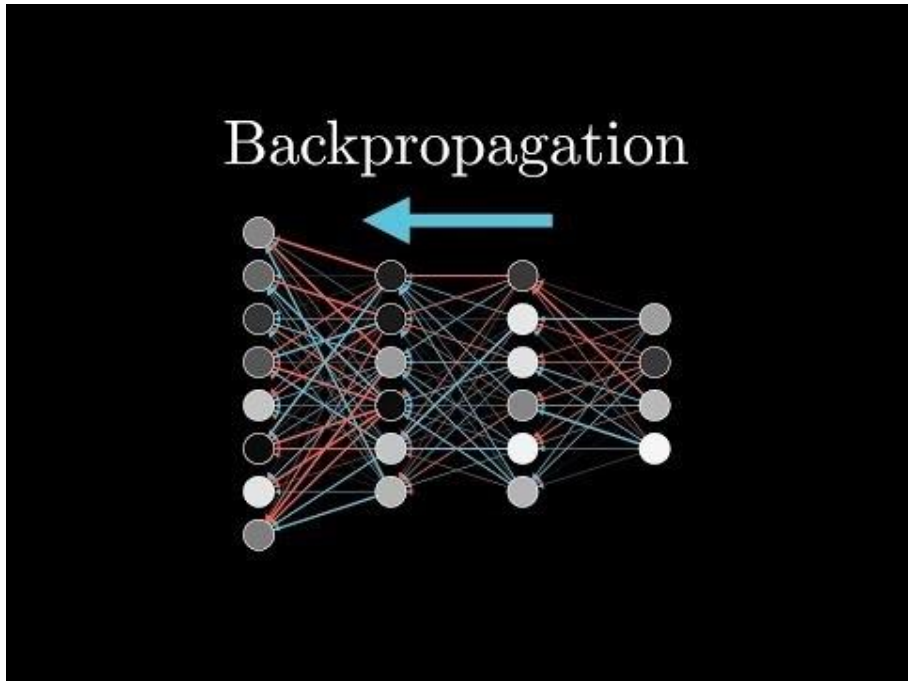
```
df = 1
dc = df
db = np.exp(b)*dc
dw0 = db
da = db
dw1 = x*da
dx = w1*da
```

Уточнения





Все еще ничего не понятно!!!!



3blue1brown

https://www.youtube.com/watch?v=Ilg3gGewQ5U

http://cs231n.github.io/optimization-2/